Booz | Allen | Hamilton®

System-Wide Safety

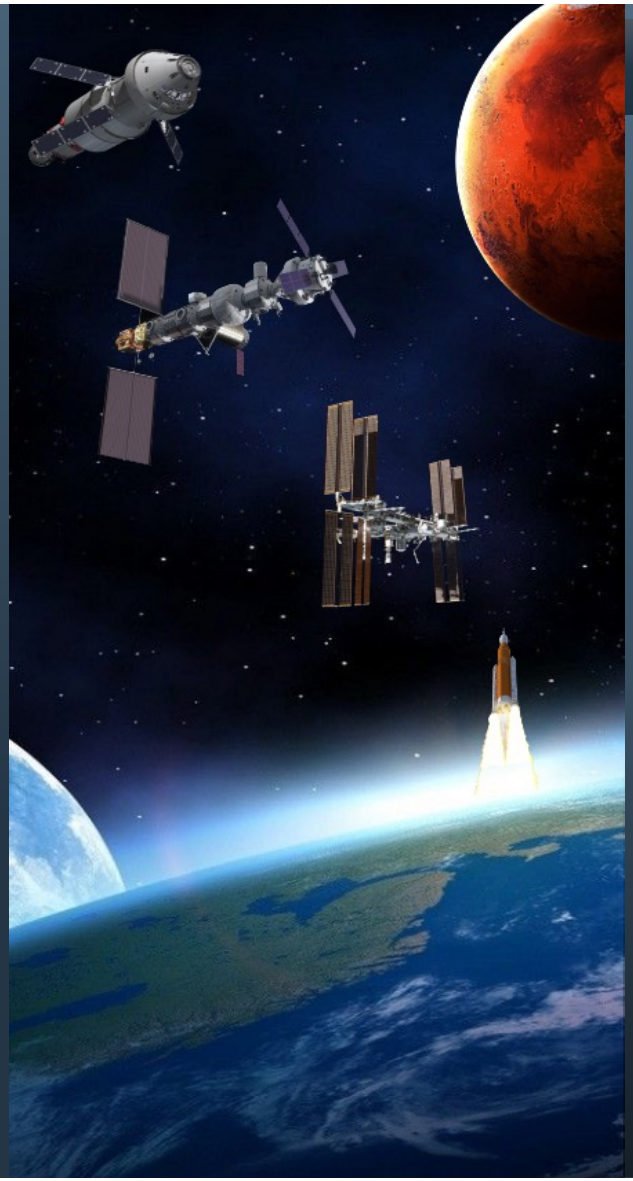
Machine Learning Methods for Advancing Risk Precursor Identification Tools in Commercial Airline Terminal Area Operations

SWS Technical Challenge #1 Integrated Terminal-Area Risk

Mr. Fasil Alemante, BAH

Mr. Chad Stephens, NASA Langley

Dr. Nikunj Oza, NASA Ames



Booz | Allen | Hamilton®

JSC Strategic Business Integration Office (XB)

Aviation Risk Precursor Identification Demonstration

Machine Learning for Advancing Risk Precursor Identification Tools Assessment

SPEAC II Task Order 004

NASA LANGLEY RESEARCH CENTER

MAR 1, 2023

CONTENTS

TEAM INTRODUCTION

Name	ROLE
Shannon Walker	➢ Task Manager / Task Lead
Fasil Alemante	➢ Data Scientist
Rami Houssami	➢ Data Scientist
Farley Reynolds	➢ Software Developer

BACKGROUND

- NASA System-Wide Safety (SWS) Project is investigating ways to integrate emerging urban air mobility (UAM) vehicle operations into the National Airspace System (NAS).
- This demonstration is intended to showcase
  - Back end and user interface proof of concept to display aviation risk precursor detection in a terminal area from System Wide Information Management (SWIM) data
- This work supports the ARMD Strategic Implementation Plan, 2019:
  - Strategic Thrust 1: Safe, Efficient Growth in Global Operations
  - Strategic Thrust 5: In-Time System-Wide Safety Assurance
  - Strategic Thrust 6: Assured Autonomy for Aviation Transformation
- This work supports SWS Technical Challenge-1
  - Milestone TC1-02-02, “Demo proof of concept showing integration of NASA & AOC capabilities”

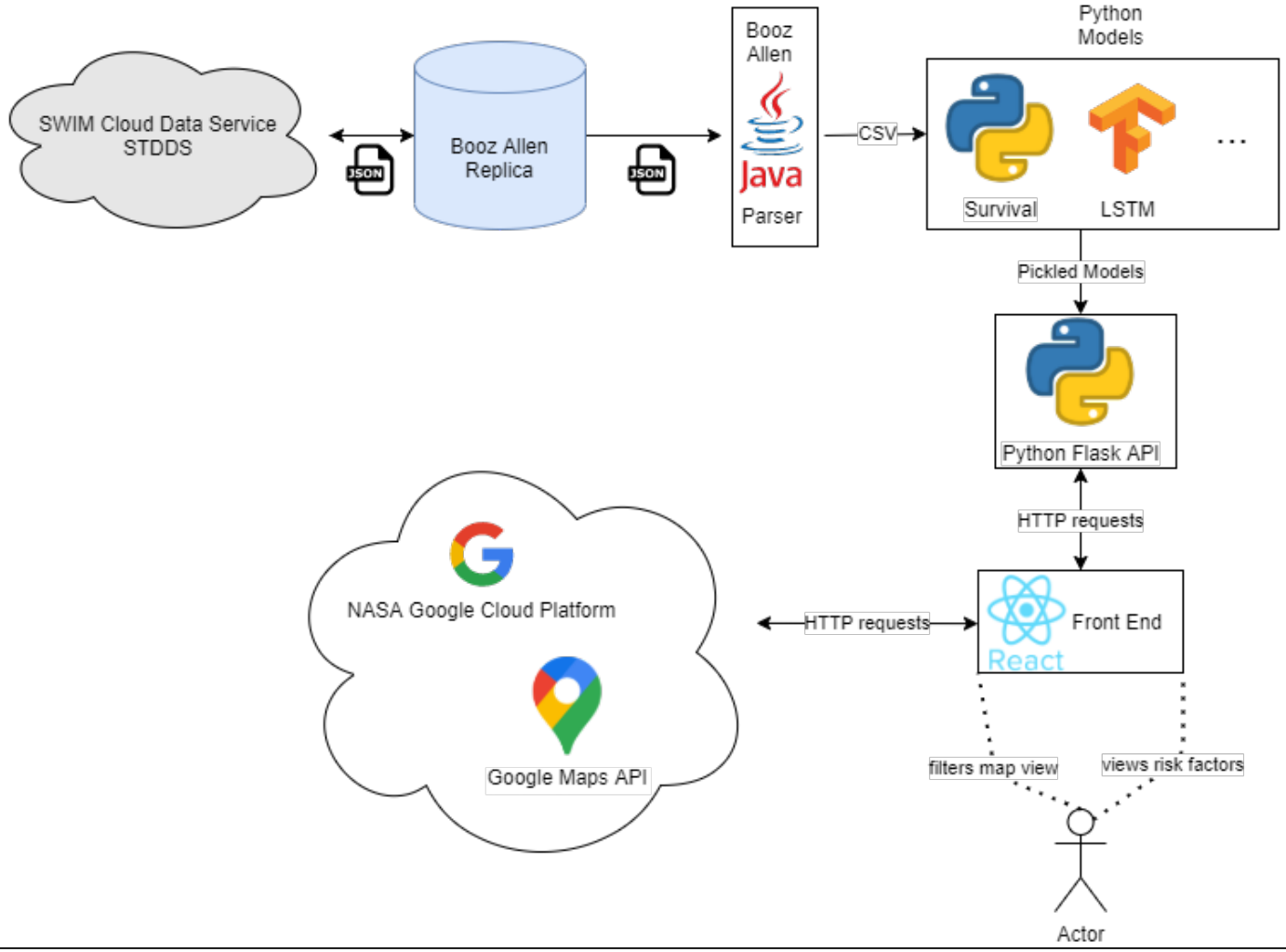
BACKGROUND

- Analyses performed thus far are meant to take advantage of initial, limited data sets.
- Goal was to find a relation that could be included and evolved into a display capability. Thus, interpretability of results is a relatively high importance feature.
- Problem statement: An unstable approach (vertical) is defined as a descent (negative vertical velocity) of greater than 1000 ft per minute (1).
- The goal is to characterize the risk of an unstable approach for a flight at the earliest possible time step to aid dispatchers.

1. https://www.skybrary.aero/tutorials/stabilised-approach

CONTENTS

HIGH-LEVEL ARCHITECTURE DIAGRAM – CURRENT STATE



ARCHITECTURE DESCRIPTION

- Back End
  - STDDS – SWIM Terminal Data Distribution System.
  - Replica – Booz Allen replica of STDDS that was used in building the proof of concept. It has historical data going back several years and is conveniently indexed.
  - Java Parser – a Java applet that converts data from SWIM's native hierarchical JavaScript Object Notation (JSON) to a data science friendly long comma-separated variable (CSV) format.
  - Python – the models used for prediction and analysis are written in Python, specifically due to the great availability of data science libraries. Libraries used include pandas, numpy, statsmodels, and Tensorflow (Keras)
- Front End
  - Python Flask application programming interface (API) – web API framework that connects the front end to the Python models
  - React.js – JavaScript library that powers data visualization and graphical interface
  - Google Maps – provides map tiles and flight path overlay

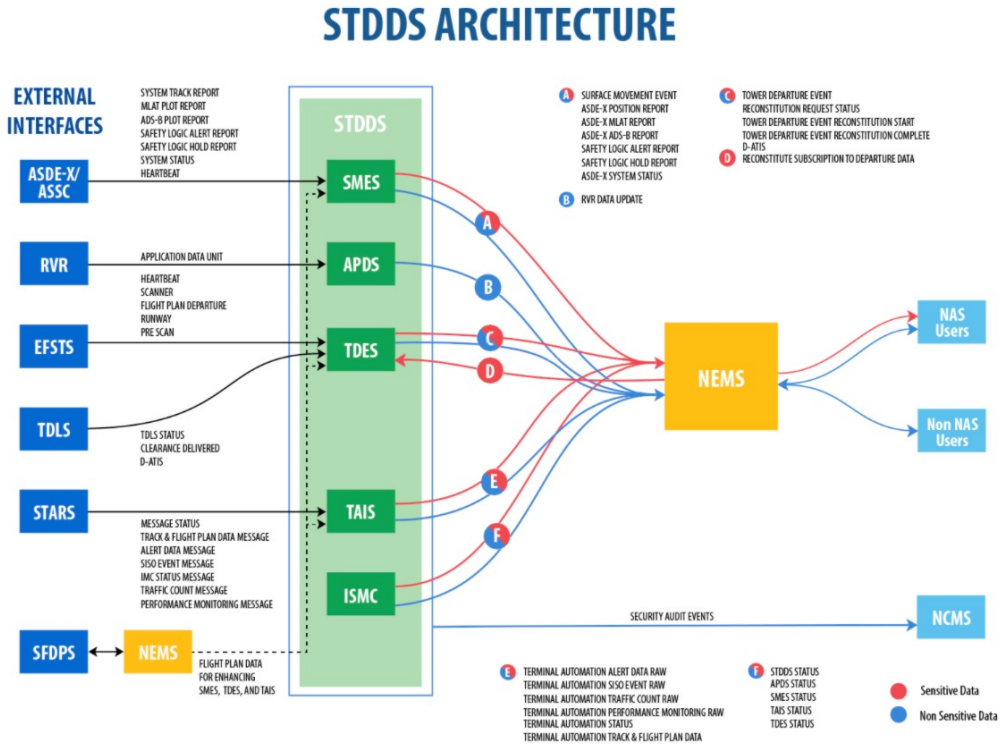
CONTENTS

DESCRIPTION OF SWIM DATA

System Wide Information Management (SWIM)

- SWIM is a Federal Aviation Administration (FAA) information-sharing platform designed to increase the sharing of data through the SWIM Cloud Distribution Service (SCDS) which contains key data sources, such as the SWIM Terminal Data Distribution System (STDDS) data outlet
- SWIM makes data available from over 200 airports, and over 400 individual systems
- STDDS interfaces with six FAA airport and terminal systems, and publishes data from them through the following services:

    1. SMES – Surface Movement Event Service
    2. APDS – Airport Data Service
    3. TDES – Tower Departure Even Service
    4. TAIS – Terminal Automation Information Service
    5. ISMC – Infrastructure, System Monitor and Control Service



Source & description of data in STDDS is available at https://www.faa.gov/air_traffic/technology/swim/stdds/

CHARACTERIZING SWIM DATA

Data Retrieval

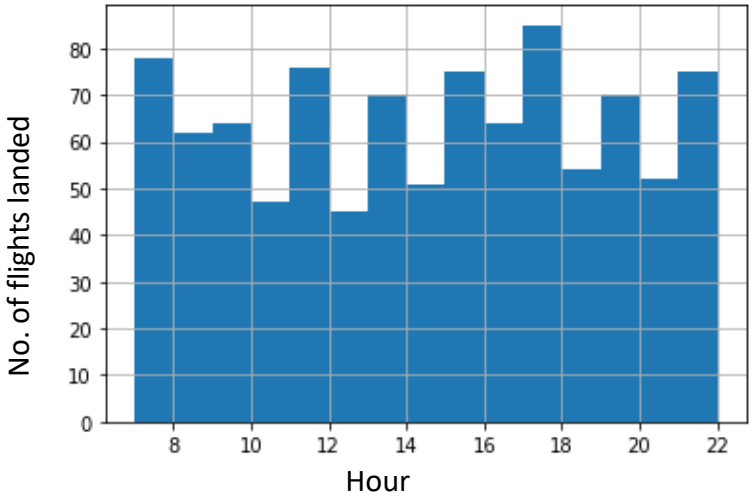
- The SWIM data originates from the SWIM Terminal Data Distribution System (STDDS)
- The data is obtained from STDDS, stored, and run through a Java parser which converts the data from a JSON (hierarchical) to CSV (flat) format

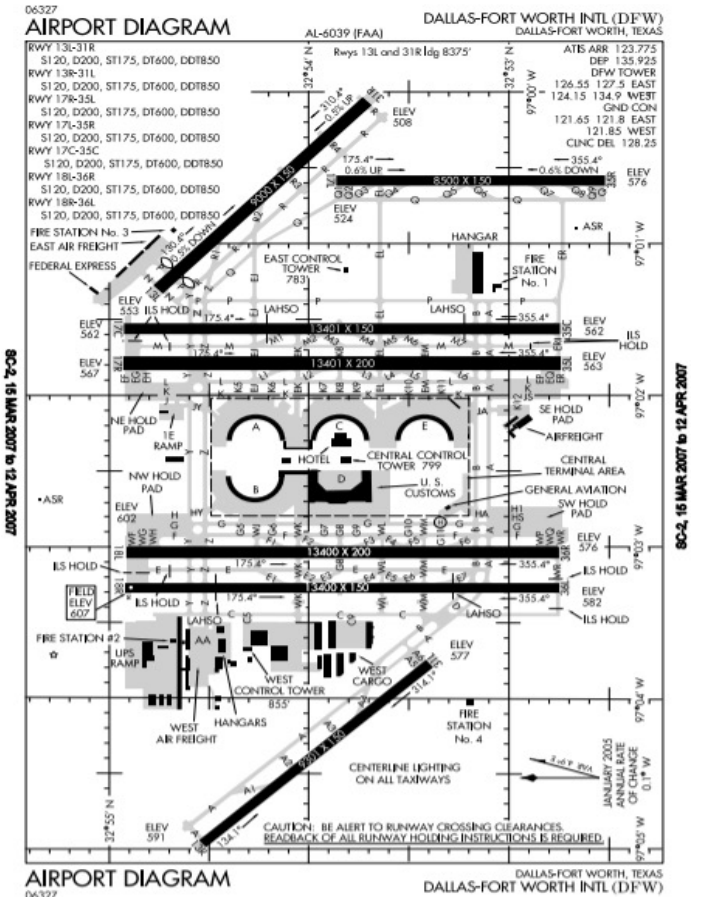
In order to use the SWIM data, a few transformations are necessary:

- Resolution for velocity is rounded to the nearest 100 ft/s. Because of this it is necessary to smooth the velocity by averaging the past two time-steps' velocities ($v_{t-1} + v_{t-2}$). Otherwise, there will be a very high number of unstable approaches.
- Velocity is provided as ft/s, so a simple conversion to ft/min is required
- SWIM provides altitude data in a fashion similar to GPS (i.e., feet above mean sea level (MSL)). Thus, to determine altitude above ground level (AGL), it is necessary to know subtract the runway's MSL elevation from SWIM's altitude.
- SWIM's aggregation of data from many systems offers access to a wide range of data and features that can provide a highly dimensional record of flight instances to be evaluated for predictive value.

CHARACTERIZING A DAY AT DALLAS-FORT WORTH (DFW)

- Data for the demonstration is a full day from the DFW runways on July 17, 2019, starting at 7 AM local time (Central Daylight Time: -5 from GMT) and ending at about 10:30 PM
- Weather: 15-20 mph winds, clear sky, low of 79°F at 9 AM to high of 94°F at 3 PM
- A total of 968 flights landed on the runways.
- Average of 46 flights landing per hour with a peak between 5 to 6 pm with over 80 aircraft landing in that hour





DFW runway layout Source: FAA

CONTENTS

TIME SERIES BASED ANALYSIS

- Considering the problem statement, the base requirement for interpretability is a prediction at a given time step. Naturally, this type of problem lends itself to time series analysis.
- Concept: flights have a descent trend. By separating the trend component from any stochastic component, the altitude at future time steps can be predicted.
- AutoRegressive, Integrated, and Moving Averages (ARIMA) (Box-Jenkins approach) are very useful methods to capture common time variant components. It allows for the use of exogenous variables as well.
- Future pivot: great for short run but degrades quickly. Does not model the actual thought process; it is only a forecasting method.
- Pros: very quick to calculate and forecast, would scale well in a production environment

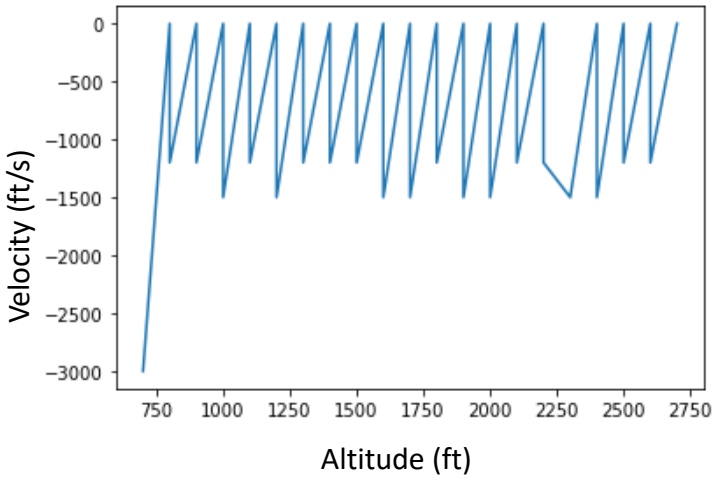


Figure: plot showing descent velocity as a time series over altitude.

SURVIVAL ANALYSIS

- Reconsidering the problem statement as “what is the likelihood of a failure state (mean time to failure), given the current conditions?”
- Answers the following question: Given what is known about previous flights up to this point (i.e., time step, altitude, and descent velocity), what is the likelihood of this flight ending with an unstable approach? Stated differently, the empirical cumulative distribution function (CDF) (Kaplan-Meier estimator) is used to determine likelihood of a future failure state.
- Since the modelling is based on initial conditions, updates can be made for more and more specific scenarios
- Pros: Very easy to interpret and visualizes extremely well. Also scales very well and can increase in accuracy with more data. Well suited to the problem space.

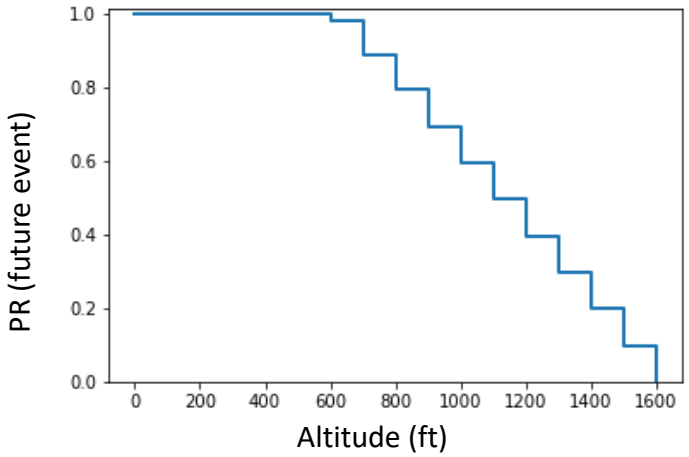


Figure: plot showing survival curve for flights. Dropoff represents the risk at that point.

LONG SHORT-TERM MEMORY (LSTM)

- Long Short-term Memory - a type of sequence-to-sequence neural network that, like survival analysis, is well suited to the problem space. It incorporates learning long order (time) dependence. Works like an autoencoder that encodes/decodes temporal relationships.
- Train on off and on nominal sample. Given the past *x* time steps, the model then predicts *y* future time steps. Based on the predictions, the model determines if the flight should be flagged for review.
- Pros: Very easy to interpret for dispatchers, as it boils down to a Boolean condition
- Cons: No answer given as to "why" the flight was flagged, which is a "black box"

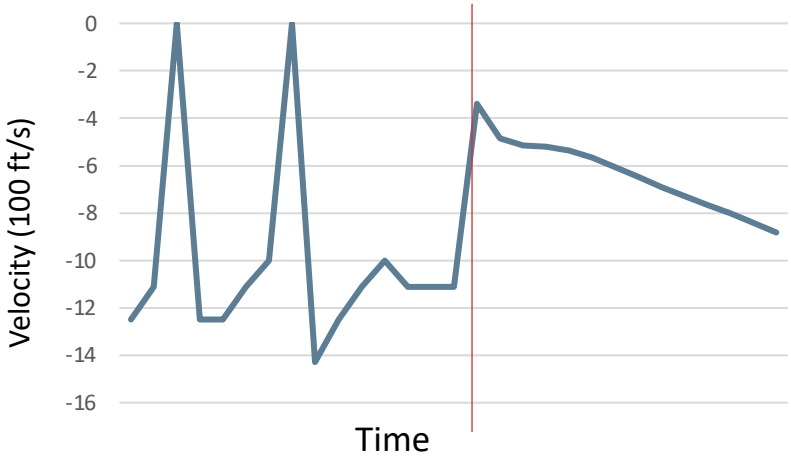


Figure: Vertical bar represents start of predicted results for this correctly predicted on nominal flight

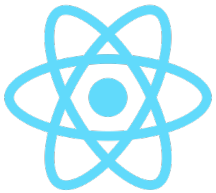
CONTENTS

VISUALIZATION BACK END: PYTHON FLASK API

- Flask is a micro web framework written in Python
  - Fairly lightweight
  - No unnecessary functionality
- Allows for quick responses (given the proper hardware) even with large datasets and computation
- Due to the extensible nature of an API the backend can support multiple models
  - Has different API calls for each model and can be extended or changed as needed for the application
- The back end currently supports two models
  - Survival Analysis
  - LSTM
- Simply send one of the API calls the appropriate data and it will return the outcome from one of the models to whatever service is calling it
  - Can be used by different services
    - User interface (UI)
    - Provide data or analytic capabilities to another service

VISUALIZATION FRONT END: REACT.JS

- A JavaScript library for building UIs
- Efficiently updates and renders components when data changes
  - Can handle complex UIs with low overhead
    - Ensures faster rendering
- Technology stack independent
  - Does not rely on a particular back end
  - If the data supplied to it matches what the UI expects then it should work
- Can use client side (user's browser) to offload tasks from any potential back end
  - Limits back end resource usage for only important tasks
  - Allows for higher user counts
- Can be employed on different platforms
  - Desktop
  - Mobile



VISUALIZATION FRONT END: GOOGLE MAPS API

- Contained in LaRC's Google Cloud Platform (GCP) instance
- Employs three main features of the Google Maps API
  - Heat Map Layer
    - Two ways to interpret based on what a model returns
      - Model returns weight or risk score:
        - Greater intensity around points that have a higher weight or risk score and areas with higher flight density
      - Model returns a classification:
        - Greater intensity only around areas with higher flight density
  - Polyline Layer (Flight Paths)
    - Given the coordinates of a flight path it can visualize where a flight has been
    - Paired with coloring, can show the outcome of a model when given a data point
      - Coloring can be used for other uses as well
  - Markers (Flight Icon/Current Location)
    - Shows where in the data the flight terminated
    - In a near real time scenario this will show the flight's current location
    - A user can click on a flight and see any information that is available for the flight
    - Paired with coloring, can show the outcome of a model at the marker's point

CONTENTS

RACE INTEGRATION

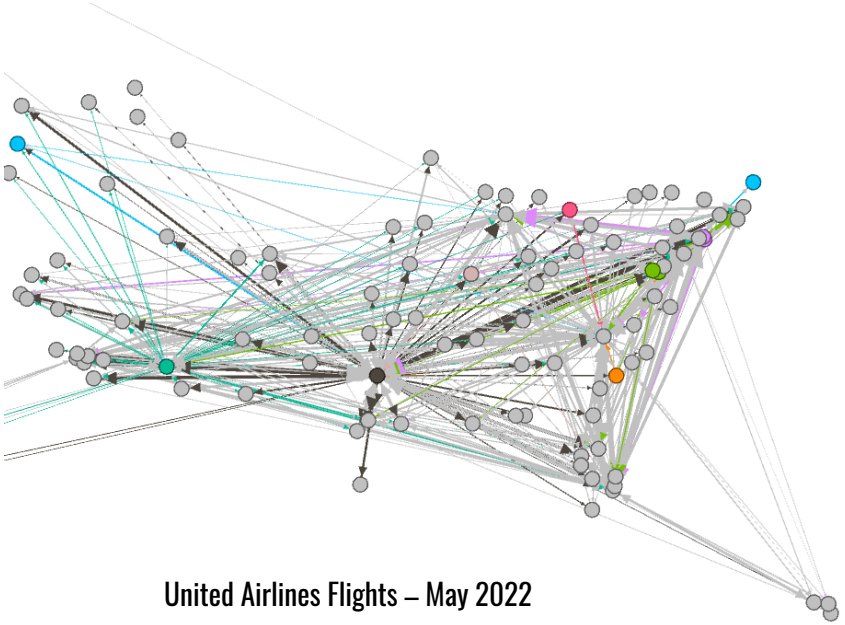
- The Runtime for Airspace Concept Evaluation (RACE) is an actor-based framework intended for simulation
- Actors
  - Every object, update, etc., constitutes an actor
  - Actors are defined in Scala files which are called upon by configuration files which tell RACE which actors to run and how
  - “Ingest actors” ingest the data; these ingest SWIM data well
  - “Archive actors” read archived data (as opposed to live, real-time data)
  - “Route actors” visualize the tracks
- Cesium is a powerful tool for visualization
  - An open platform for software applications for 3D data
  - A supported application with libraries
- For the current demo
  - Repurposed a replay actor that was built around ADS-B CSV files to work with the SWIM data
  - SWIM ingest actors already exist for backend level access to SWIM
  - Used a route actor to display in Cesium
- Possible future avenues
  - Opportunities to write functionality to display things differently
  - Write an actor that interacts with the ARPI algorithm via an API

CONTENTS

APPLIED GRAPH AND NETWORK ALGORITHMS

The network-nature of the National Airspace System (NAS) makes it an ideal candidate for graph analytics and algorithms, whereby:

Airports → Nodes
Flight routes → Edges



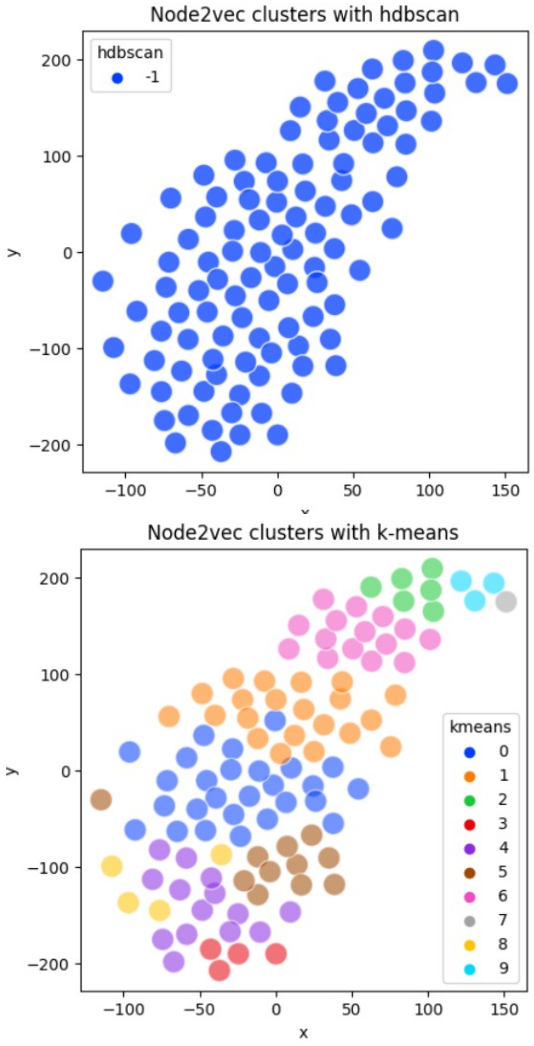
United Airlines Flights – May 2022

APPLIED GRAPH AND NETWORK ALGORITHMS

Graph Machine Learning is the process of applying machine learning techniques on network structures and relationships.

Node2vec is an algorithm for encoding graphical data into a latent or vector space. This enables various vector operations for:

- Clustering
- Similarity search
- Inference tasks



APPLIED GRAPH AND NETWORK ALGORITHMS

Further analysis:

Validation of any node2vec model and embedding representations stands in the use case.

For data analysis we can pose the questions

”do cosine similarities of airports produce incites?”

“are cascade-potentials captured in the vector representation?”

```
model.wv.most_similar('DEN')
```

```
[('ONT', 0.9978517889976501),
 ('BUR', 0.9978493452072144),
 ('SNA', 0.9978044033050537),
 ('DTW', 0.9977736473083496),
 ('ABQ', 0.9977695345878601),
 ('SLC', 0.9977158904075623),
 ('CVG', 0.9977013468742371),
 ('GRR', 0.9977009296417236),
 ('COS', 0.997681200504303),
 ('OMA', 0.9976595640182495)]
```

BACKUP SLIDES

REPLICA AND PARSER DETAILS

- Why use a replica of STDDS?
  - Using this existing replica cleared some technical hurdles early in the project, as the data was already available and did not need any rework to filter needed information. In future versions, this abstraction won't be necessary, and any data source that can provide inputs resembling the current data can be incorporated.
- What is the Java parser?
  - This applet was developed by Booz Allen for a previous investment and converts the hierarchical JSON format to a more data science friendly CSV format. Just as using the STDDS replica saved resources, using this existing parser locally on Booz Allen laptops also accelerated progress.
- Why build with Python?
  - Python is the industry standard for using thoroughly matured machine learning libraries which offer robust open-source modules that span nearly all phases of a machine learning cycle.
  - It can be run in virtually any environment and is highly integrable with other systems.

CURRENT/FUTURE AVENUES

- In addition to the work already done, investigating new and promising ideas are being investigated
  - Merging LSTM and survival analysis to create a best of breed system
    - Found some prior art that incorporates both, which looks promising
    - Deep Recurrent Survival Analysis: https://arxiv.org/pdf/1809.02403.pdf (Ren et al., 2018)
  - Incorporating higher fidelity flight data
    - A NASA contractor supporting ARC-TI recommended Automatic Dependent Surveillance – Broadcast (ADSB) (GPS from aircraft) and provided documentation on this data stream in ARC-TI's Runtime for Airspace Concept Evaluation (RACE)
    - Reviewed DASHlink data supplied by CSAOB, and built a basic parser to ingest and make .MAT files (a propriety format for MATLAB) available to the Python models. Data source: https://c3.nasa.gov/dashlink/projects/85/
    - Feature engineering